METHODS AND SYSTEMS FOR PROCESSING EMAIL MESSAGES

FIELD OF THE INVENTION

[0001] The invention generally relates to search engines. More particularly, the invention relates to methods and systems for processing email messages.

BACKGROUND OF THE INVENTION

[0002] Users generate and access a large number of articles, such as emails, web pages, word processing documents, spreadsheet documents, instant messenger messages, and presentation documents, using a client device, such as a personal computer, personal digital assistant, or mobile phone. Some articles are stored on one or more storage devices coupled to, accessible by, or otherwise associated with the client device(s). Users sometimes wish to search the storage device(s) for articles.

[0003] Conventional client-device search applications may significantly degrade the performance of the client device. For example, certain conventional client-device search applications typically use batch processing to index all articles, which can result in noticeably slower performance of the client device during the batch processing. Additionally, batch processing occurs only periodically. Therefore, when a user performs a search, the most recent articles are sometimes not included in the results. Moreover, if the batch processing is scheduled for a time when the client device is not operational and

is thus not performed for an extended period of time, the index of articles associated with the client device can become outdated. Conventional client-device search applications can also need to rebuild the index at each batch processing or build new partial indexes and perform a merge operation that can use a lot of client-device resources. Conventional client-device search applications also sometimes use a great deal of system resources when operational, resulting in slower performance of the client device.

[0004] Additionally, conventional client-device search applications can require an explicit search query from a user to generate results, and may be limited to examining file names or the contents of a particular application's files.

SUMMARY

[0005] Embodiments of the present invention comprise methods and systems for processing email messages. In one embodiment, the occurrence of a condition is determined indicating at least one email message transfer of an email message by an email application, wherein determining the occurrence of the condition is external to the email application, the email message is identified, wherein the email message comprises event data, an email event is compiled from at least some of the event data, and the email event is indexed. Determining the condition can comprise one or more of determining if files associated with the email application have been updated, determining if an email related operating system condition has occurred, and determining if a packet received

from a network comprises an email protocol. In another embodiment, emails can be associated with a conversation.

[0006] These exemplary embodiments are mentioned not to limit or define the invention, but to provide examples of embodiments of the invention to aid understanding thereof. Exemplary embodiments are discussed in the Detailed Description, and further description of the invention is provided there. Advantages offered by the various embodiments of the present invention may be further understood by examining this specification.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] These and other features, aspects, and advantages of the present invention are better understood when the following Detailed Description is read with reference to the accompanying drawings, wherein:

[0008] Figure 1 is a diagram illustrating an exemplary environment in which one embodiment of the present invention may operate;

[0009] Figure 2 is a flow diagram illustrating an exemplary method of capturing and processing event data associated with a client device in one embodiment of the present invention;

[0010] Figure 3 is a flow diagram illustrating an exemplary method of capturing an email event in one embodiment of the present invention; and

[0011] Figure 4 is a flow diagram illustrating an exemplary method of indexing an email event in one embodiment of the present invention.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

[0012] Referring now to the drawings in which like numerals indicate like elements throughout the several figures, Figure 1 is a block diagram illustrating an exemplary environment for implementation of an embodiment of the present invention. While the environment shown in Figure 1 reflects a client-side search engine architecture embodiment, other embodiments are possible. The system 100 shown in Figure 1 includes multiple client devices 102a-n that can communicate with a server device 150 over a network 106. The network 106 shown in Figure 1 comprises the Internet. In other embodiments, other networks, such as an intranet, may be used instead. Moreover, methods according to the present invention may operate within a single client device that does not communicate with a server device or a network.

[0013] The client devices 102a-n shown in Figure 1 each include a computer-readable medium 108. The embodiment shown in Figure 1 includes a random access memory (RAM) 108 coupled to a processor 110. The processor 110 executes computer-executable program instructions stored in memory 108. Such processors may include a

microprocessor, an ASIC, state machines, or other processor, and can be any of a number of suitable computer processors, such as processors from Intel Corporation of Santa Clara, California and Motorola Corporation of Schaumburg, Illinois. Such processors include, or may be in communication with, media, for example computer-readable media, which stores instructions that, when executed by the processor, cause the processor to perform the steps described herein. Embodiments of computer-readable media include, but are not limited to, an electronic, optical, magnetic, or other storage or transmission device capable of providing a processor, such as the processor 110 of client 102a, with computer-readable instructions. Other examples of suitable media include, but are not limited to, a floppy disk, CD-ROM, DVD, magnetic disk, memory chip, ROM, RAM, an ASIC, a configured processor, all optical media, all magnetic tape or other magnetic media, or any other medium from which a computer processor can read instructions. Also, various other forms of computer-readable media may transmit or carry instructions to a computer, including a router, private or public network, or other transmission device or channel, both wired and wireless. The instructions may comprise code from any suitable computer-programming language, including, for example, C, C++, C#, Visual Basic, Java, Python, Perl, and JavaScript.

[0014] Client devices 102a-n can be coupled to a network 106, or alternatively, can be stand alone machines. Client devices 102a-n may also include a number of external or internal devices such as a mouse, a CD-ROM, DVD, a keyboard, a display device, or other input or output devices. Examples of client devices 102a-n are personal computers,

digital assistants, personal digital assistants, cellular phones, mobile phones, smart phones, pagers, digital tablets, laptop computers, Internet appliances, and other processor-based devices. In general, the client devices 102a-n may be any type of processor-based platform that operates on any suitable operating system, such as Microsoft® Windows® or Linux, capable of supporting one or more client application programs. For example, the client device 102a can comprise a personal computer executing client application programs, also known as client applications 120. The client applications 120 can be contained in memory 108 and can include, for example, a word processing application, a spreadsheet application, an email application, an instant messenger application, a presentation application, an Internet browser application, a calendar/organizer application, a video playing application, an audio playing application, an image display application, a file management program, an operating system shell, and other applications capable of being executed by a client device. Client applications may also include client-side applications that interact with or accesses other applications (such as, for example, a web-browser executing on the client device 102a that interacts with a remote e-mail server to access e-mail).

[0015] The user 112a can interact with the various client applications 120 and articles associated with the client applications 120 via various input and output devices of the client device 102a. Articles include, for example, word processor documents, spreadsheet documents, presentation documents, emails, instant messenger messages, database entries, calendar entries, appointment entries, task manager entries, source code

files, and other client application program content, files, messages, items, web pages of various formats, such as HTML, XML, XHTML, Portable Document Format (PDF) files, and media files, such as image files, audio files, and video files, or any other documents or items or groups of documents or items or information of any suitable type whatsoever.

[0016] The user's 112a interaction with articles, the client applications 120, and the client device 102a creates event data that may be observed, recorded, analyzed or otherwise used. An event can be any occurrence possible associated with an article, client application 120, or client device 102a, such as inputting text in an article, displaying an article on a display device, sending an article, receiving an article, manipulating an input device, opening an article, saving an article, printing an article, closing an article, opening a client application program, closing a client application program, idle time, processor load, disk access, memory usage, bringing a client application program to the foreground, changing visual display details of the application (such as resizing or minimizing) and any other suitable occurrence associated with an article, a client application program, or the client device whatsoever. Additionally, event data can be generated when the client device 112a interacts with an article independent of the user 112a, such as when receiving an email or performing a scheduled task.

[0017] The memory 108 of the client device 102a can also contain a capture processor 124, a queue 126, and a search engine 122. The client device 102a can also contain or is in communication with a data store 140. The capture processor 124 can

capture events and pass them to the queue 126. The queue 126 can pass the captured events to the search engine 122 or the search engine 122 can retrieve new events from the queue 126. In one embodiment, the queue 126 notifies the search engine 122 when a new event arrives in the queue 126 and the search engine 122 retrieves the event (or events) from the queue 126 when the search engine 122 is ready to process the event (or events). When the search engine receives an event it can be processed and can be stored in the data store 140. The search engine 122 can receive an explicit query from the user 112a or generate an implicit query and it can retrieve information from the data store 140 in response to the query. In another embodiment, the queue is located in the search engine 122. In still another embodiment, the client device 102a does not have a queue and the events are passed from the capture processor 124 directly to the search engine 122. According to other embodiments, the event data is transferred using an information exchange protocol. The information exchange protocol can comprise, for example, any suitable rule or convention facilitating data exchange, and can include, for example, any one of the following communication mechanisms: Extensible Markup Language – Remote Procedure Calling protocol (XML/RPC), Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP), shared memory, sockets, local or remote procedure calling, or any other suitable information exchange mechanism.

[0018] The capture processor 124 can capture an event by identifying and compiling event data associated with an event. Examples of events include sending or receiving an email message, a user viewing a web page, saving a word processing document, printing

a spreadsheet document, inputting text to compose or edit an email, opening a presentation application, closing an instant messenger application, entering a keystroke, moving the mouse, and hovering the mouse over a hyperlink. An example of event data captured by the capture processor 124 for an event involving the receipt of an email message by the user 112a can comprise the sender of the message, the recipients of the message, the time and date the message was received, and the content of the message.

[0019] In the embodiment shown in Figure 1, the capture processor 124 comprises multiple capture components. For example, the capture processor 124 shown in Figure 1 comprises a separate capture component for each client application in order to capture events associated with each application. The capture processor 124 can also comprise a separate capture component that monitors overall network activity in order to capture event data associated with network activity, such as the receipt or sending of an instant messenger message. The capture processor 124 shown in Figure 1 also can comprise a separate client device capture component that monitors overall client device performance data, such as processor load, idle time, disk access, the client applications in use, and the amount of memory available. The capture processor 124 shown in Figure 1 also comprises a separate capture component to monitor and capture keystrokes input by the user and a separate capture component to monitor and capture items, such as text, displayed on a display device associated with the client device 102a. An individual capture component can monitor multiple client applications and multiple capture components can monitor different aspects of a single client application.

[0020] In one embodiment, the capture processor 124, through the individual capture components, can monitor activity on the client device and can capture events by a generalized event definition and registration mechanism, such as an event schema. Each capture component can define its own event schema or can use a predefined one. Event schemas can differ depending on the client application or activity the capture component is monitoring. Generally, the event schema can describe the format for an event, for example, by providing fields for event data associated with the event (such as the time of the event) and fields related to any associated article (such as the title) as well as the content of any associated article (such as the document body). An event schema can describe the format for any suitable event data that relates to an event. For example, an event schema for an email message event received by the user 112a can include the sender, the recipient or list of recipients, the time sent, the date sent, and the content of the message. An event schema for a web page currently being viewed by a user can include the Uniform Resource Locator (URL) of the web page, the time being viewed, and the content of the web page. An event schema for a word processing document being saved by a user can include the title of the document, the time saved, the format of the document, the text of the document, and the location of the document. More generally, an event schema can describe the state of the system around the time of the event. For example, an event schema can contain a URL for a web page event associated with a previous web page that the user navigated from. In addition, event schema can describe fields with more complicated structure like lists. For example, an event schema can

contain fields that list multiple recipients. An event schema can also contain optional fields so that an application can include additional event data if desired. An event schema can also contain location information as described above.

[0021] The capture processor 124 can capture events occurring presently (or "real-time events") and can capture events that have occurred in the past (or "historical events"). Real-time events can be "indexable" or "non-indexable". In one embodiment, the search engine 122 indexes indexable real-time events, but does not index non-indexable real-time events. The search engine 122 may determine whether to index an event based on the importance of the event. Indexable real-time events can be more important events associated with an article, such as viewing a web page, loading or saving a file, and receiving or sending an instant message or email. Non-indexable events can be deemed not important enough by the search engine 122 to index and store the event, such as moving the mouse or selecting a portion of text in an article. Non-indexable events can be used by the search engine 122 to update the current user state. While all real-time events can relate to what the user is currently doing (or the current user state), indexable real-time events can be indexed and stored in the data store 140. Alternatively, the search engine 122 can index all real-time events. Real-time events can include, for example, sending or receiving an article, such as an instant messenger message, examining a portion of an article, such as selecting a portion of text or moving a mouse over a portion of a web page, changing an article, such as typing a word in an email or pasting a sentence in a word processing document, closing an article, such as

closing an instant messenger window or changing an email message being viewed, loading, saving, opening, or viewing an article, such as a word processing document, web page, or email, listening to or saving an MP3 file or other audio/video file, or updating the metadata of an article, such as book marking a web page, printing a presentation document, deleting a word processing document, or moving a spreadsheet document.

[0022] Historical events are similar to indexable real-time events except that the event occurred before the installation of the search engine 122 or was otherwise not captured, because, for example, the search engine 122 was not operational for a period of time while the client device 102a was operational or because no capture component existed for a specific type of historical event at the time the event took place. Examples of historical events include the user's saved word processing documents, media files, presentation documents, calendar entries, and spreadsheet documents, the emails in a user's inbox, and the web pages book-marked by the user. The capture processor 124 can capture historical events by periodically crawling the memory 108 and any associated data storage device for events not previously captured by the capture processor 124. The capture processor 124 can also capture historical events by requesting certain client applications, such as a web browser or an email application, to retrieve articles and other associated information. For example, the capture processor 124 can request that the web browser application obtain all viewed web pages by the user or request that the email application obtain all email messages associated with the user. These articles may not currently exist in memory 108 or on a storage device of the client device 102a. For

example, the email application may have to retrieve emails from a server device. In one embodiment, the search engine 122 indexes historical events.

[0023] In the embodiment shown in Figure 1, events captured by the capture processor 124 are sent to the queue 126 in the format described by an event schema. The capture processor 124 can also send performance data to the queue 126. Examples of performance data include current processor load, average processor load over a predetermined period of time, idle time, disk access, the client applications in use, and the amount of memory available. Performance data can also be provided by specific performance monitoring components, some of which may be part of the search engine 122, for example. The performance data in the queue 126 can be retrieved by the search engine 122 and the capture components of the capture processor 124. For example, capture components can retrieve the performance data to alter how many events are sent to the queue 126 or how detailed the events are that are sent (fewer or smaller events when the system is busy) or how frequently events are sent (events are sent less often when the system is busy or there are too many events waiting to be processed). The search engine 122 can use performance data to determine when it indexes various events and when and how often it issues implicit queries.

[0024] In one embodiment, the queue 126 holds events until the search engine 122 is ready to process an event or events. Alternatively, the queue 126 uses the performance data to help determine how quickly to provide the events to the search engine 122. The

queue 126 can comprise one or more separate queues including a user state queue and an index queue. The index queue can queue indexable events, for example. Alternatively, the queue 126 can have additional queues or comprise a single queue. The queue 126 can be implemented as a circular priority queue using memory mapped files. The queue can be a multiple-priority queue where higher priority events are served before lower priority events, and other components may be able to specify the type of events they are interested in. Generally, real-time events can be given higher priority than historical events, and indexable events can be given higher priority than non-indexable real-time events. Other implementations of the queue 126 are possible. In another embodiment, the client device 102a does not have a queue 126. In this embodiment, events are passed directly from the capture processor 124 to the search engine 122.In other embodiments, events can be transferred between the capture components and the search engine using suitable information exchange mechanisms such as: Extensible Markup Language – Remote Procedure Calling protocol (XML/RPC), Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP), shared memory, sockets, local or remote procedure calling, or any other suitable information exchange mechanism.

[0025] The search engine 122 can contain an indexer 130, a query system 132, and a formatter 134. The query system 132 can retrieve real-time events and performance data from the queue 126. The query system 132 can use performance data and real-time events to update the current user state and generate an implicit query. An implicit query can be an automatically generated query based on the current user state. The query

system 132 can also receive and process explicit queries from the user 112a. Performance data can also be retrieved by the search engine 122 from the queue 126 for use in determining the amount of activity possible by the search engine 122.

[0026] In the embodiment shown in Figure 1, indexable real-time events and historical events (indexable events) are retrieved from the queue 126 by the indexer 130. Alternatively, the queue 126 may send the indexable events to the indexer 130. The indexer 130 can index the indexable events and can send them to the data store 140 where they are stored. The data store 140 can be any type of computer-readable media and can be integrated with the client device 102a, such as a hard drive, or external to the client device 102a, such as an external hard drive or on another data storage device accessed through the network 106. The data store can be one or more logical or physical storage areas. In one embodiment, the data store 140 can be in memory 108. The data store 140 may facilitate one or a combination of methods for storing data, including without limitation, arrays, hash tables, lists, and pairs, and may include compression and encryption. In the embodiment shown in Figure 1, the data store comprises an index 142, a database 144 and a repository 146.

[0027] In one embodiment, when the indexer 130 receives an event, the indexer 130 can determine, from the event, terms (if any) associated with the event, the time of the event (if available), images (if any) associated with the event, and/or other information defining the event. The indexer 130 can also determine if the event relates to other

events and associate the event with related events. For example, for a received email event, the indexer 130 can associate the email event with other message events from the same conversation or string. The emails from the same conversation can be associated with each other in a conversation object, which can be stored in the data store 140.

[0028] The indexer 130 can send and incorporate the terms and times, associated with the event in the index 142 of the data store 140. The event can be sent to the database 144 for storage and the content of the associated article and any associated images can be stored in the repository 146. The conversation object associated with email messages can be stored in the database 144.

[0029] In the embodiment shown in Figure 1, a user 112a can input an explicit query into a search engine interface displayed on the client device 102a, which is received by the search engine 122. The search engine 122 can also generate an implicit query based on a current user state, which can be determined by the query system 132 from real-time events. Based on the query, the query system 132 can locate relevant information in the data store 140 and provide a result set. In one embodiment, the result set comprises article identifiers for articles associated with the client applications 120 or client articles. Client articles include articles associated with the user 112a or client device 102a, such as the user's emails, word processing documents, instant messenger messages, previously viewed web pages and any other article or portion of an article associated with the client device 102a or user 112a. An article identifier may be, for example, a Uniform Resource

Locator (URL), a file name, a link, an icon, a path for a local file, or other suitable information that may identify an article. In another embodiment, the result set also comprises article identifiers for articles located on the network 106 or network articles located by a search engine on a server device. Network articles include articles located on the network 106 not previously viewed or otherwise referenced by the user 112a, such as web pages not previously viewed by the user 112a.

[0030] The formatter 134 can receive the search result set from the query system 132 of the search engine 122 and can format the results for output to a display processor 128. In one embodiment, the formatter 134 can format the results in XML, HTML, or tab delineated text. The display processor 128 can be contained in memory 108 and can control the display of the result set on a display device associated with the client device 102a. The display processor 128 may comprise various components. For example, in one embodiment, the display processor 128 comprises a Hypertext Transfer Protocol (HTTP) server that receives requests for information and responds by constructing and transmitting Hypertext Markup Language (HTML) pages. In one such embodiment, the HTTP server comprises a scaled-down version of the Apache Web server. The display processor 128 can be associated with a set of APIs to allow various applications to receive the results and display them in various formats. The display APIs can be implemented in various ways, including as, for example, DLL exports, COM interface, VB, JAVA, or .NET libraries, or a web service.

[0031] Through the client devices 102a-n, users 112a-n can communicate over the network 106, with each other and with other systems and devices coupled to the network 106. As shown in Figure 1, a server device 150 can be coupled to the network 106. In the embodiment shown in Figure 1, the search engine 122 can transmit a search query comprised of an explicit or implicit query or both to the server device 150. The user 112a can also enter a search query in a search engine interface, which can be transmitted to the server device 150 by the client device 102a via the network 106. In another embodiment, the query signal may instead be sent to a proxy server (not shown), which then transmits the query signal to server device 150. Other configurations are also possible.

[0032] The server device 150 can include a server executing a search engine application program, such as the Google™ search engine. In other embodiments, the server device 150 can comprise a related information server or an advertising server. Similar to the client devices 102a-n, the server device 150 can include a processor 160 coupled to a computer-readable memory 162. Server device 150, depicted as a single computer system, may be implemented as a network of computer processors. Examples of a server device 150 are servers, mainframe computers, networked computers, a processor-based device, and similar types of systems and devices. The server processor 160 can be any of a number of computer processors, such as processors from Intel Corporation of Santa Clara, California and Motorola Corporation of Schaumburg, Illinois. In another embodiment, the server device 150 may exist on a client-device. In still another embodiment, there can be multiple server devices 150.

[0033] Memory 162 contains the search engine application program, also known as a network search engine 170. The search engine 170 can locate relevant information from the network 106 in response to a search query from a client device 102a. The search engine 170 then can provide a result set to the client device 102a via the network 106. The result set can comprise one or more article identifiers. An article identifier may be, for example, a Uniform Resource Locator (URL), a file name, a link, an icon, a path for a local file, or anything else that identifies an article. In one embodiment, an article identifier can comprise a URL associated with an article.

[0034] In one embodiment, the server device 150, or related device, has previously performed a crawl of the network 106 to locate articles, such as web pages, stored at other devices or systems coupled to the network 106, and indexed the articles in memory 162 or on another data storage device. It should be appreciated that other methods for indexing articles in lieu of or in combination with crawling may be used, such as manual submission.

[0035] It should be noted that other embodiments of the present invention may comprise systems having different architecture than that which is shown in Figure 1. For example, in some other embodiments of the present invention, the client device 102a is a stand-alone device that is not permanently coupled to a network. The system 100 shown in Figure 1 is merely exemplary, and is used to explain the exemplary methods shown in Figures 2-4.

[0036] Various methods in accordance with embodiments of the present invention may be carried out. For example, in one embodiment, the occurrence of a condition is determined indicating at least one email message transfer of an email message by an email application, wherein determining the occurrence of the condition is external to the email application, the email message is identified, wherein the email message comprises event data, an email event is compiled from at least some of the event data, and the email event is indexed. The email message transfer can comprise receiving the email message or sending the email message. Event data can comprise at least one of sender data, a date and time associated with the event, content from the email message, and a conversation ID. The email event and the email message can be stored. The email application can comprise a client-based email application and/or a network-based email application.

[0037] In one embodiment, determining the occurrence of the condition comprises determining if files associated with the email application have been updated. In another embodiment, determining the occurrence of the condition comprises determining if an email related operating system condition has occurred. The operating system condition can comprise an email icon output on a display associated with the client device, an email message box output on a display associated with the client device, and determining metadata for an email indicator associated with the email message displayed in the email application. In another embodiment, determining the occurrence of the condition comprises determining if a packet received from a network comprises an email protocol. For a network-based email application, determining the occurrence of the condition can,

in one embodiment, comprise analyzing a web page associated with the network-based email application. In another embodiment determining the occurrence of the condition may comprise determining an email protocol and an email server based on analysis of settings associated with the email application or network traffic, and periodically polling the email server for new email messages.

[0038] In one embodiment, the email event is associated with a conversation. Associating the email message with a conversation can comprises determining if an existing conversation relevant to the email event exists, associating the email event with an existing conversation if the existing conversation is determined to be relevant to the email event, and associating the email event with a new conversation if no existing conversation is determined to exist that is relevant to the email event. Determining if an existing conversation relevant to the email event exists may comprise an analysis of the event data associated with the email event. The analysis of the event data may comprise analysis of one or more of email message subject, date, content, sender and recipients. Determining if an existing conversation relevant to the email event exists may comprise determining a conversation ID associated with the email message.

[0039] Figure 2 illustrates an exemplary method 200 that provides a method for capturing and processing an event. This exemplary method is provided by way of example, as there are a variety of ways to carry out methods according to the present invention. The method 200 shown in Figure 2 can be executed or otherwise performed

by any of various systems. The method 200 is described below as carried out by the system 100 shown in Figure 1 by way of example, and various elements of the system 100 are referenced in explaining the example method of Figure 2.

[0040] In 202, the capture processor 124 captures an event. The event can be a real-time event or can be a historical event. The capture processor 124 can capture a real-time event by identifying and compiling event data associated with the event upon the occurrence of the event. The capture processor 124 can capture a historical event, for example, by periodically crawling the memory 108 or associated data storage device of the client device 112a for previously uncaptured articles or receiving articles or data from client applications and identifying and compiling event data associated with the event. The capture processor 124 may have separate capture components for each client application, network monitoring, performance data capture, keystroke capture, and display capture. In one embodiment, the capture component can use a generalized event definition mechanism, such as an event schema that it has previously defined and registered with the client device 102a, to capture or express the event.

[0041] Figure 3 illustrates an exemplary method 202 for capturing an email event involving an email message transfer by an email application. A message transfer can comprise the receipt of an email message or the sending of an email message. Figure 3 illustrates an exemplary method for capturing an email event involving the receipt of an email message. It will be appreciated by those skilled in the art that this method can also

apply to email messages that are sent, in addition to, or in lieu of email messages that are received. In 302, the occurrence of a condition is identified indicating the receipt of an email message by an email application on the client device 112a. Some conventional email applications have an application program interface (API) that can allow a capture component access to the email application to determine if an email has been received, to determine if an email has been sent, or to determine if other events associated with the application have taken place. Other conventional email applications do not provide APIs that can allow a capture component 124 to monitor the receipt of incoming email messages or sending of email messages through the email application. In this situation, the capture component 124 can determine the occurrence of a condition indicating an email message transfer, such as, for example, the receipt of an email message, without the use of an API or other express indication of an email message transfer from the email application. Therefore, the capture component 124 can determine the occurrence of a condition indicating an email message transfer external to or independent of the email application.

[0042] In one embodiment, the email capture component can periodically crawl files associated with the email application to determine if the files or other data such as registry entries have been updated since the last crawl. For example, a conventional email application can have one or more files associated with each email message folder. Each file can contain the email messages in the associated folder. The files can contain information, such as date and time, associated with the last update of the file. The email

capture component can compare the current update information associated with each file with update information from a previous crawl of the files and determine if each file has been updated. If one or more of the files have been updated, this can act as a condition independent of the email application indicating that an email or emails have been received by the email application since the last crawl. Identifying that a file has been updated may be done by periodically checking metadata associated with the file such as the last update time or the size of the file, or may be done by registering with the operating system to receive notification when certain file changes occur. The update information may be that the update time for a file has been changed, subsequent analysis of the file may determine that a new message has been received, or that no new message has been received (for example, a message may have been deleted).

[0043] Alternatively the capture component can periodically inspect other resources associated with the email application. For example, the capture component can inspect registry entries or watch for operating system notifications regarding registry entries associated with the email application.

[0044] In another embodiment, the email capture component can monitor the operating system for conditions indicating the receipt of an email. The email capture component can, for example, monitor display calls by the operating system indicating the receipt of an email. For example, some operating systems can output an icon for display in a system tray indicating that an email message has been received by the email

application. Identifying an email icon may comprise, for example, identifying a window with a specific identifier, such as a class name, or analyzing the pixels displayed on a display device. An operating system can also output a message window indicating that an email has been received by the email application. If an email application is open, metadata associated with email indicators can indicate that a new email message has been received. For example, newly received emails can be indicated in a bold font on the display of the email application. The email capture component can monitor display calls to determine the metadata associated with email message indicators. These operating system actions can act as a condition independent of the email application that indicates that an email has been received by the email application. In another embodiment, an operating system capture component can monitor and determine the operating system actions and upon determining a condition associated with the email application, the operating system capture component can indicate the occurrence of such condition to the email capture component.

[0045] In a further example, the email capture component can monitor packets received (or sent) by the client device 112a from (or to) the network 106 and can identify an email packet by the protocol used, for example POP and IMAP. For example, the email capture component can monitor incoming packets and when the capture component determines that an incoming packet is in an email protocol this can act as a condition independent of the email application that an email has been or will soon be received by the email application. Outgoing email messages may be monitored in a similar fashion.

In another embodiment, a network capture component can monitor and determine the receipt of an email protocol packet and upon determining receipt of such packet, the network capture component can indicate the occurrence of such condition to the email capture component or actually capture the email content itself.

[0046] In another embodiment, the capture component can identify an email protocol, such as IMAP or POP, based on analysis of email client application settings or analysis of network traffic, and can subsequently determine and poll an email server, such as an IMAP or POP server, periodically to check for new mail. If new mail is received the capture component may retrieve the mail from the server without removing it or marking it as read (so the email application later retrieves it), or may instruct the email application to retrieve the message. This may allow the capture component to identify new mail faster than by waiting for the email application to check the email server. For example, the email application may not check for new mail unless the user activates the application.

[0047] In a further example, the email capture component can monitor HTML displayed in a network-based email application system such as Yahoo™ Mail or Hotmail™. Analysis of the HTML can identify a string that indicates new mail is available, or can identify new mail itself, such as mail listed in bold font in a listing of messages. This analysis can be done by using a browser plug-in such as a Browser Helper Object (BHO) or by analyzing the contents of the application window.

[0048] After the identification of a condition indicating the receipt of an email message, in 304, the email capture component can identify a new received email message or messages. In one embodiment, where the email application has an API that allows the email capture component to have access to the email application to determine the receipt of new email, the new email message can be identified by the email application.

[0049] In another embodiment, where the email capture component can determine the receipt of a new email or email independent of the email application, the email capture component can determine new received email messages. In this embodiment, the email capture component can identify a new email message by crawling the files associated with an email application. The email capture component can first determine which file or files have been updated since the last crawl as explained above (if this has not been done). For the file or files that have been updated, the email capture component can determine the new received email message or messages based on the date and time the email message was received by the email application. For example, the date and time of the last captured email previously captured by the capture component can be compared to the date and time associated with the email messages in the file(s). An email message with an associated date and time after the date and time of the last captured email message can be identified by the email capture component as a new email message.

[0050] In 306, the identified email event is captured. In one embodiment, the email capture component can capture the email event by compiling event data associated with

the email event. In one embodiment, the email capture component can use an event schema to express the event data. For example, the event schema for an email event can include one or a combination of sender information, recipient information, time that the email message was received, a date that the email message was received, the subject of the email message and the content of the email message. The event schema can also contain a conversation ID. A conversation ID can indicate the particular conversation (sometimes referred to as a "thread") that the email message is associated with. The conversation ID can be provided by the email application or can be determined by the email capture component. In one embodiment, an email message can contain an identification number or numbers associated with other message(s) that the email message is related to, forming a chain of relationships that constitute a conversation.

[0051] In another embodiment, the conversation ID is determined based on analysis of one or more of the message subject, date, content, and author or recipients. In one embodiment, groups of messages with the same subject are given the same conversation ID, with the following refinements. Typically, certain reply indicators such as "Re:", "Fwd:", "Fw:", prepended to the subject. may be used as indicators that the email is part of a previous thread. The text of the subject after these reply indicators (the normalized subject) can then be compared to previous normalized subjects. If the original subject has no reply indicators, the email may be part of a new conversation, and thus it may receive a new conversation ID. If there is no overlap between the authors and other recipients of the email messages, then email messages with the same normalized subject may be given

different conversation IDs. For example, if person A sends person B and person C an email message with subject X, and the person D sends person B and person E an email message also with subject X, then these email messages and any subsequent replies may have separate conversation IDs, as the participants (other than the person whose email is being indexed) do not overlap. In another embodiment, if there is a gap of greater than a certain amount of time between messages, the subsequent messages are given a new conversation ID. In another embodiment, quoted text in the email can be compared to emails in potential threads to determine if they are related.

[0052] In another embodiment "related" conversations can be determined by inspecting the email message body and the email attachments.

[0053] Some email applications include the headers of the "related" email message(s) in the body of the current email message, while other email applications attach the "related" email message(s). In certain instances, the user may attach relevant email messages or threads to a new message. Extracting this information can allow the search engine 122 to provide the user with both the email conversation thread as well as "related" mail messages or conversations.

[0054] Returning to Figure 2, in 204, the capture processor 124 determines whether the event captured is an indexable event. As explained above, some real-time events may not be indexed (non-indexable real-time events). In one embodiment, non-indexable real-time events are used to update the current user state and are, for example, examining

a portion of an article, changing an article, and closing an article. In this embodiment, non-indexable events are not indexed or sent for storage by the indexer 130. Indexable events can be indexable real-time events or historical events. The receipt of an email message can be an indexable event.

[0055] If an indexable event is determined, then, in 206, the event can be sent by the capture processor 124 to the queue 126 with an indication that it is an indexable event. In the embodiment shown, indexable real-time events are sent to both a user state queue and an index queue within queue 126 and historical events are sent to the index queue within the queue 126. Alternatively, indexable real-time events may not be sent to the user state queue to save computational time. The capture processor 124 can send the event in a form described by an event schema to the queue 126. If the event is determined to be a non-indexable event, then, in 206, the non-indexable event can be sent by the capture processor 124 to the user state queue of the queue 126 with an indication that it is not to be indexed.

[0056] In one embodiment, the queue 126 holds the event until the search engine is ready to receive it. Based on the event data, the event can be prioritized on the queue 126 for handling. For example, historical events are given a lower priority for processing by the queue 126 than real-time events. In one embodiment, when the indexer 130 is ready to process another event, it can retrieve an event or events from the index queue in the queue 126. The query system 132 can retrieve an event or events from the user state

queue of the queue 126, when it is ready to update the user state. In another embodiment, a queue is not used and events are sent directly to the search engine 122 from the capture processor 124.

[0057] In 208, the indexer 130 indexes and stores the event. The indexer 130 can retrieve an event from the queue 126 when it is ready to process the event. In one embodiment, the indexer 130 determines if the event is a duplicate event and if not assigns an Event ID to the event. The indexer 130 can also associate the event with related events. In the embodiment shown in Figure 2, the indexer determines indexable terms associated with the event, dates and times associated with the event, and other data associated with the event from the event schema. The indexer 130 can associate the Event ID with the indexable terms that are contained in the index 142. The event can be stored in the database 144 and the content of the event can be stored in the repository 146.

[0058] Figure 4 illustrates an exemplary method 210 for indexing and storing an email event. In 402, the indexer 130 retrieves an email event from the queue 126 and determines that the event is an email event. In one embodiment, the indexer 130 can determine if the retrieved email event is a duplicate of a previously processed email event, and if not can assign the email event an event ID. The indexer 130 can also parse out indexable terms, including one or more of sender and recipient names, subject, message body and attachments, times and dates associated with the event, and a

conversation ID from the event data, which can include the content of the email message and attachments, advertisements or other information associated with the email message.

[0059] In 404, the indexer 130 determines whether the email event is associated with a new conversation or if the email event is associated with an existing conversation. The indexer 130 can compare the conversation ID associated with the email event with previously indexed conversation IDs to make this determination.

[0060] If the indexer 130 determines that the email event is associated with a new conversation, then the indexer 130, in 406, can create a new conversation object. A conversation object can associate an email event with related email events from the same conversation.

[0061] If the indexer 130 determines that the email event is associated with an existing conversation, then the indexer 130, in 408, can load the relevant conversation object from the database 144 of the data store 140. By associating email events with a conversation, a user is able to search for and retrieve complete email conversations. Also the indexer 130 can determine "related" messages and conversations, different than the current conversations, and create associations between them.

[0062] Once the event has been associated with a conversation object and provided with a conversation object identifier, the email event can be stored in the data store 140. In 410, the email message can be stored in the repository 146. In 412, the event and conversation object can be stored in the database 144. In 414, the indexable terms from

the email can be stored in the index 142. In one embodiment, the Event ID from the event is associated with terms in the index 142 that equate to the indexable terms of the event.

[0063] The capturing of email events that are indexed and stored by the search engine 122 allows the user 112a to search for and retrieve email messages on the client device 102a and allows the search engine to automatically search for email messages on the client device 102a.

[0064] The environment shown reflects a client-side search engine architecture embodiment. Other embodiments are possible, such as a stand-alone client device or a network search engine.

[0065] Similarly, while in some portions of the description, exemplary embodiments are described with reference to receipt of email messages, it will be appreciated from this foregoing description that the invention may in various embodiments process incoming and/or outgoing email messages.

[0066] Furthermore, the invention should not be limited to client-based or network-based (e.g., web-based or browser-based) email applications, but may be implemented in a variety of email applications, including client-based, web/network-based, or a combination client-network email application.

[0067] While the above description contains many specifics, these specifics should not be construed as limitations on the scope of the invention, but merely as exemplifications of the disclosed embodiments. Those skilled in the art will envision many other possible variations that are within the scope of the invention.